REMARKS/ARGUMENTS

1**. Reply to the Rejections under 35 U.S.C. 102(e)**

In paragraph 3 on page 2 of the Official Action, claims 1-4, 10, 11, 15, 22, 32, 33-36, 44, 45, 49, 58, and 61-73 were rejected under 35 U.S.C. 102(e) as being anticipated by Burns et al. U.S. 6,925,515 B2. Applicants respectfully traverse. Applicants respectfully submit that Burns et al. does not identically show every element of claims 1-4, 10, 11, 15, 22, 32, 33-36, 44, 45, 49, 58, and 61-73.

"For a prior art reference to anticipate in terms of 35 U.S.C. § 102, every element of the claimed invention must be identically shown in a single reference." Diversitech Corp. v. Century Steps, Inc., 7 U.S.P.Q.2d 1315, 1317 (Fed. Cir. 1988), quoted in In re Bond, 910 F.2d 831,15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990) (vacating and remanding Board holding of anticipation; the elements must be arranged in the reference as in the claim under review, although this is not an *ipsis verbis* test).

Applicants' claim 1 calls for a network file server computer responding to a concurrent write request from a client by performing a sequence of seven steps. The network file server computer responds by:

(a) obtaining a lock for the file; and then

(b) preallocating a metadata block for the file; and then

(c) releasing the lock for the file; and then

(d) asynchronously writing to the file in the data storage; and then

(e) obtaining the lock for the file; and then

(f) committing the metadata block to the file in the data storage; and then

(g) releasing the lock for the file.

The applicants' drawings show these steps (a) to (g) in FIG. 11 in boxes 101, 102, 103, 104-105, 106, 107, and 108, respectively, as described in applicants' specification on page 27 lines 3-23.

In applicants' view, Burns does not disclose a network file server computer responding to a concurrent write request from a client by performing the applicants' claimed sequence of seven steps. Instead, Burns discloses a network file server computer that responds to concurrent write requests by performing a different sequence of steps depending on the "locking mode" associated with the request. The citation in page 2 of the Official Action to Burns col. 10 lines 14-30, however, indicates that the Official Action has identified the "locking mode" and the kind of write operation in Burns that is most pertinent to the subject matter of applicants' claim 1.

Burns (FIG. 3) discloses a Client/Server distributed file system on a storage area network (SAN). (Col. 5, lines 64-65.) The metadata is stored separately from the data. The metadata is stored in a file server cluster. The data is stored on shared disks of the SAN. (Col. 7, lines 1-9.) In a preferred embodiment, the clients of the distributed file system include a DBMS server, and a plurality of web servers. The DBMS server may update a web page, and the web servers may read the web page as the update occurs. The DBMS server obtains an exclusive "producer lock" on the web page. The producer lock enables the holder to write data, and allocate and cache data for an "out of place" write that writes the data to a different physical storage location than from which it was read. By performing an "out-of-place" write, the old data still exists and is

available to clients. Once the writer completes the write and releases the producer lock, the previous data is invalidated and the clients are informed of the new location of the data. Clients can then read the new data from storage when needed, and the server reclaims the old data blocks. (Burns, col. 5 lines 31-54; col. 10 lines 13-29.)

Page 3 of the Official Action cites Burns col. 10 lines 14-30 for applicants' step (b) of preallocation, and also for applicants' step (c) of releasing the lock for the file, and for applicants' step (d) for asynchronously writing to the file. However, Burns col. 10 lines 14-30 do not disclose that the lock for the file (obtained in the first step (a)) is released after step (b) and before step (d), nor do Burns col. 10 lines 14-30 disclose that the lock for the file is obtained again in step (e) and released again in step (g), as recited in applicants' claim 1. Moreover, Burns col. 10 lines 36-38 expressly disclose that multiple concurrent writers need only a guarantee that the data does not change location; hence allocation requires an exclusive lock. Furthermore, Burns as a whole teaches that the "out of place" write requires an exclusive producer lock, because the "out of place" write changes the allocation. (See Burns, col. 10, lines 30-47.) Moreover, when clients are writing data that changes the length of the file, they hold whole file locks. (Burns, col. 9, lines 27-30.) Thus, Burns teaches or suggests that the "out of place" write would include:

a) obtaining a lock for the file;

b) obtaining allocations of data blocks for the new data;

c) asynchronously writing new data to the allocated data blocks;

d) committing the allocated data blocks to the file to the data storage; and

e) releasing the lock for the file.

In comparison to applicants' claim 1, there is no disclosure or suggestion in Burns of releasing the lock for the file after allocation of the data blocks (e.g., by preallocating a metadata block for the file as more specifically defined in applicants' claim 1) and before asynchronously writing the new data to the file, nor is there any disclosure of again obtaining the lock on the file after writing the new data to the file and before committing the allocated data blocks (e.g., including committing the metadata block as more specifically defined in applicants' claim 1) to the file in the data storage. In addition, Burns does not disclose details of how a block of data is allocated to the file, or committed to the file. For example, there is nothing in Burns disclosing that a metadata block is preallocated for the file, and then the file is asynchronously written to, and then the metadata block is committed to the file in data storage, as recited in applicants' claim 1.

Burns teaches away from releasing the lock for the file in step (c) and then asynchronously writing to the file in step (d) and then again obtaining the lock for the file in step (e), because Burns, as set out above, teaches that the file would be exclusively write-locked by the producer lock when a write operation adding a block to the file is performed on the file. See, for example, Burns col. 11, lines 47-49: "With C and P locks, the web servers are not updated until the P lock holder releases the lock, generally on closing the file."

With respect to applicants' dependent claims 2 and 3, page 3 of the Official Action cites Burns column 10 line 14 through column 11 line 14. However, Burns does not disclose details of this metadata structure such as an indirect block of metadata.

With respect to applicants' dependent claim 4, page 3 of the Official Action cites Burns, column 10 line 14 through column 11 line 14. However, Burns does not disclose concurrent writing for more than one client to a metadata block for the file. Instead, as discussed above, Burns discloses that a write operation that would change the allocation of a block requires an exclusive lock.

With respect to applicants' dependent claim 10, page 3 of the Official Action cites Burns column 10 line 14 through column 11 line 14. However, Burns does not disclose writing a metadata block to a log in storage of the network file server for committing the metadata block.

With respect to applicants' dependent claim 11, page 4 of the Official Action cites Burns column 10 line 14 through column 11 line 14. Burns, however, does not disclose gathering together preallocated metadata blocks for a plurality of client write requests to the file, and committing together the preallocated metadata blocks for the plurality of client write requests to the file by obtaining the lock for the file, committing the gathered preallocated metadata blocks for the plurality of client write requests to the file, and then releasing the lock for the file. Instead, if a client obtains a producer lock for a change in allocation of the file, this producer lock is exclusive and prevents other clients from obtaining a producer lock on the file. (See col. 10, lines 50-65.) The producer lock would be obtained and held before and while any new data blocks, and their metadata, would be preallocated and later committed. Therefore Burns teaches away from gathering together preallocated metadata blocks for a plurality of client write requests to the file, and committing together the preallocated metadata blocks for the plurality of client write requests to the file by obtaining the lock for the file, committing the gathered preallocated

metadata blocks for the plurality of client write requests to the file, and then releasing the lock for the file.

With respect to applicants' independent claim 15, as discussed above with reference to applicants' claim 11, it is respectfully submitted that Burns fails to disclose gathering together preallocated metadata blocks for a plurality of client write requests to the file, and committing together the preallocated metadata blocks for the plurality of client write requests to the file by obtaining the lock for the file, committing the gathered preallocated metadata blocks for the plurality of client write requests to the file, and then releasing the lock for the file.

With respect to applicants' independent claim 32, see applicants' remarks above with reference to applicants' claim 1. In short, Burns does not disclose details of how a new metadata block is added to the file. Therefore, applicants respectfully submit that Burns does not disclose that new metadata bocks are added to the file by the applicants' recited sequence of the six steps (b), (c), (d), (e), (f), and (g).

With respect to claims 33-36, applicants respectfully traverse for the reasons given above with reference to claims 1-4.

With respect to claims 44-45, applicants respectfully traverse for the reasons given above with reference to claim 10-11.

With respect to independent claim 49, applicants respectfully traverse for the reasons given above with reference to claim 15.

With respect to independent claim 58, applicants respectfully traverse for the reasons given above with reference to claim 32.

With respect to independent claim 61, Burns discloses that a change in allocation requires an exclusive lock. Burns, col. 10, lines 35-37. In Burns, there can be multiple concurrent writers that write directly to the data storage of the SAN, thereby bypassing the server cluster. In this case, however, Database and Parallel Application Locking (instead of File System Locking) is used to ensure data consistency. See the table at the top of column 10 in Burns. Thus, Burns does not disclose that the network file server includes both an uncached write interface and a cached write interface in which the network file server is further programmed to invalidate cache blocks in a file system cache including sectors being written to by an uncached write interface. Instead, Burns teaches a different way of ensuring consistency between client caches and the result of an "out-of-place" write that changes the allocation of the file. The "out-of-place" writer obtains an exclusive "producer lock" on the file, and then writes to the file. "The writer must release the P lock on close to publish the file. The server sends location updates to all clients that hold consumer locks 410. The clients immediately invalidate the affected blocks in their cache." (Burns, column 11, lines 34-38.) Thus, Burns teaches the network file server invalidating client caches when file system locking is used, rather than the network file server invalidating a file system cache of a cached read-write interface when an uncached write interface bypasses the file system cache by performing a sector-aligned write operation.

2. Reply to the Rejections under 35 U.S.C. 103(a)

In paragraph 5 on page 7 of the Official Action, claims 5-9, 12-14, 16-21, 23-28, 37-43, 46-48, and 50-54 were rejected under 35 U.S.C. 103(a) as being unpatentable over Burns in view of Marcotte U.S. 6,449,614 B1. In response, applicants respectfully traverse. In short, as discussed above, various elements of the respective base claims are missing from Burns, and Marcotte does not disclose these elements missing from Burns, so that the applicants' claimed invention does not result from the proposed combination of Burns and Marcotte. Moreover, the applicants' claims define a substantial improvement over Burns and Marcotte by providing a new way of handling a write request that changes the allocation of the data blocks to a file, and this new way of handling such a write request solves a long-felt need for reducing contention during multi-threaded or multi-processor access to a shared file system.

Applicants respectfully submit that Burns does not address the problem of providing concurrent writes for the case of a write operation that changes the allocation of a block of data. Instead, Burns col. 10 lines 35-37 says: "The only guarantee multiple concurrent writers need is that the data does not change location, hence allocation requires an exclusive lock." Thus, applicants still rely on their previously submitted evidence of a long-felt but unsolved problem that in a shared or parallel file system, the potential speed at which an application may execute is impaired by the need for file locking.

As previously argued by applicants, Chang et al. US 2005/0039049 is evidence of a long-felt but unsolved problem that in a shared or parallel file system, the potential speed at which an application may execute is impaired by the need for file locking. Chang teaches that for "an

application having its own serialization or locking mechanisms" (Chang, paragraph [0014]), "multiple processes may write to the same block of data within the file at approximately the same time as long as they are not changing the allocation of the block of data, i.e. either allocating the block, deallocating the block of data, or changing the block of data." (Chang, paragraph [0015].) The applicants' invention further solves this problem by enabling multiple processes to write to the file at approximately the same time when updating the metadata structure associated with the file. The applicants cited additional evidence (on the IDS form filed on April 15, 2009) showing that shared or parallel file systems and their associated problems have been known for at least a decade prior to the filing of the applicants' patent application.

Marcotte discloses an interface system and methods for asynchronously updating a share resource with locking facility. Tasks make updates requested by calling tasks to a shared resource serially in a first come first served manner, atomically, but not necessarily synchronously, such that a current task holding an exclusive lock on the shared resource makes the updates on behalf of one or more calling tasks queued on the lock. Updates waiting in a queue on the lock to the shared resource may be made while the lock is held, and others deferred for post processing after the lock is released. Some update requests may also, at the calling application's option, be executed synchronously. Provision is made for nested asynchronous locking. Data structures (wait_elements) describing update requests may be queued in a wait queue for update requests awaiting execution by a current task, other than the calling task, currently holding an exclusive lock on the shared resource. Other queues are provided for queuing data structures removed from the wait queue but not yet processed; data structures for

requests to unlock or downgrade a lock; data structures for requests which have been processed and need to be returned to free storage; and data structures for requests that need to be awakened or that describe post processing routines that are to be run while the lock is not held. (Abstract.)

With respect to applicants' dependent claims 5-9, 12, 16-21, 23-24, 37-43, and 50-54, these claims depend from the independent claims 1, 15, 33, and 49. Burns has been distinguished above with respect to the independent claims 1, 15, 33, and 49, and Marcotte does not provide the limitations of these independent claims that are missing from Burns. Therefore the dependent claims 5-9, 12, 16-21, 23-24, 37-43, and 50-54 are patentable over the proposed combination of Burns and Marcotte. It is respectfully submitted that it would not have been obvious for one of ordinary skill to combine Burns and Marcotte in the fashion as proposed in the Official Action and then modify that combination by adding the missing limitations.

Applicants' dependent claim 5 adds to claim 1 the limitations of "wherein the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method further includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon failing to find an indication of a conflict with a concurrent write to the new block, preallocating the new block, copying at least a portion of the original block of the file to the new block, and performing the partial write to the new block." Applicants' partial block conflict queue 73 is shown in applicants' FIG. 4 and described in applicant's specification on page 15 lines 17-22 as follows:

> The preallocation method allows concurrent writes to indirect blocks within the same file. Multiple writers can write to the same indirect block tree concurrently without improper replication of the indirect blocks. Two different indirect blocks will not be allocated for replicating the same indirect block. The write threads use the partial block conflict queue 73 and the partial write wait queue 74 to avoid conflict during partial block write operations, as further described below with reference to FIG. 13.

See also applicants' FIG. 13 and specification page 28 line 22 to page 29 line 7; and FIG. 17 and page 34 line 7 to page 35 line 5.

With respect to applicants' dependent claim 5, pages 8-9 of the Official Action recognizes that Burns fails to explicitly recite the limitations added by dependent claim 5, and says that Marcotte teaches these limitations, citing column 13, lines 35 through col. 14, line 43. However, Marcotte column 13, lines 35 through col. 14, line 43 deals with managing an I/O device holding queue above a device for queuing pending I/O's if the number of I/O's issued to the device exceeds a threshold that is adjustable by an application. It is not seen where Marcotte discloses a partial write to a new block that has been copied at least in part from an original block of the file. Nor is it seen where Marcotte discloses checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon failing to find an indication of a conflict with a concurrent write to the new block, preallocating the new block, copying at least a portion of the original block of the file to the new block, and performing the partial write to the new block.

"[R]ejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." In re Kahn, 441 F. 3d 977, 988 (Fed. Cir. 2006). A fact finder should be aware of the distortion caused by hindsight bias and must be cautious of arguments reliant upon ex post reasoning. See KSR International Co. v. Teleflex Inc., 550 U.S. __, 82 USPQ2d 1385 (2007)), citing Graham, 383 U. S. at 36 (warning against a "temptation to read into the prior art the teachings of the invention in issue" and instructing courts to "guard against slipping into the use of hindsight.").

Applicants' claim 6 is similar to claim 5 in that it also adds to claim 1 the limitations of wherein the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method further includes checking a partial block conflict queue for a conflict with a concurrent write to the new block. With respect to dependent claim 6, pages 8-9 of the Official Action also recognizes that Burns fails to explicitly recite the limitations added by dependent claim 6, and says that Marcotte teaches these limitations, again citing Marcotte column 13, lines 35 through col. 14, line 43. However, Marcotte column 13, lines 35 through col. 14, line 43 deals with managing an I/O device holding queue above a device for queuing pending I/O's if the number of I/O's issued to the device exceeds a threshold that is adjustable by an application. It is not seen where Marcotte discloses a partial write to a new block that has been copied at least in part from an original block of the file. Nor is it seen where Marcotte discloses checking a partial block conflict queue for a conflict with a concurrent write to the new block.

With respect to applicants' dependent claim 12, applicants respectfully submit that Burns does not further teach checking whether a previous commit is in progress after asynchronously writing to the file and before obtaining the lock for the file for committing the metadata block to the file. As discussed above with reference to applicants' claim 1, in response to a concurrent write request from a client, Burns does not again obtain the lock for the file after the asynchronously writing to the file. Nor dos Marcotte column 12, line 7 through column 13, line 32 disclose these limitations missing from Burns or specifically deal with committing "metadata blocks" to a file.

Applicants' independent claim 13 recites "wherein the asynchronous writing to the file includes a partial write to a new block that has been copied at least in part from an original block of the file, and wherein the method includes checking a partial block conflict queue for a conflict with a concurrent write to the new block, and upon finding an indication of a conflict with a concurrent write to the new block, waiting until resolution of the conflict with the concurrent write to the new block, and then performing the partial write to the new block." Thus, Applicants' independent claim 13 is patentable over Burns in combination with Marcotte for the reasons given above with reference to applicants' claim 5.

Applicants' dependent claim 14 is dependent upon claim 13 and therefore is also patentable over Burns in combination with Marcotte for the reasons given above with reference to applicants' claim 13.

Applicants' dependent claim 16 adds to claim 15 the express limitations of claim 12 and therefore is patentable over Burns in combination with Marcotte for the reasons given above with reference to applicants' claims 12 and 15.

Applicants' dependent claim 17 adds to claim 15 the limitations of "wherein the network file server includes disk storage containing a file system, and a file system cache storing data of blocks of the file, and the method further includes the network file server computer responding to concurrent write requests by writing new data for specified blocks of the file to the disk storage without writing the new data for the specified blocks of the file to the file system cache, and invalidating the specified blocks of the file in the file system cache." Pages 13-14 of the Official Action recognize that Burns fails to explicitly recite the recited operations of "the network file server computer responding to concurrent write requests by writing new data for specified blocks of the file to the disk storage without writing the new data for the specified blocks of the file to the file system cache, and invalidating the specified blocks of the file in the file system cache." Page 14 of the Official Action cites Marcotte column 12 line 7 through column 13, line 32. However, it is not understood how the applicants' specific claim limitations are disclosed in Marcotte column 12 line 7 through column 13, line 32.

Applicants' dependent claim 18 adds to claim 17 the limitations of "the network file server computer responding to read requests for file blocks not found in the file system cache by reading the file blocks from the file system in disk storage and then checking whether the file blocks have become stale due to concurrent writes to the file blocks, and writing to the file system cache a file block that has not become stale, and not writing to the file system cache a file block that has become stale." Page 14 of the Official Action again cites Marcotte column 12 line 7 through column 13, line 32. However, it is not understood how the applicants' specific claim limitations are disclosed in Marcotte column 12 line 7 through column 13, line 32.

Applicants' dependent claim 19 adds to claim 18 the limitations of "the network file server computer checking a read-in-progress flag for a file block upon finding that the file block is not in the file system cache, and upon finding that the read-in-progress flag indicates that a prior read of the file block is in progress from the file system in the disk storage, waiting for completion of the prior read of the file block from the file system in the disk storage, and then again checking whether the file block is in the file system cache." Page 14-15 of the Official Action again cites Marcotte column 12 line 7 through column 13, line 32. However, it is not understood how the applicants' specific claim limitations are disclosed in Marcotte column 12 line 7 through column 13, line 32.

Applicants' dependent claim 20 adds to claim 18 the limitations of "the network file server computer setting a read-in-progress flag for a file block upon finding that the file block is not in the file system cache and then beginning to read the file block from the file system in disk storage, clearing the read-in-progress flag upon writing to the file block on disk, and inspecting the read-in-progress flag to determine whether the file block has become stale due a concurrent write to the file block." Page 15 of the Official Action again cites Marcotte column 12 line 7 through column 13, line 32. However, it is not understood how the applicants' specific claim limitations are disclosed in Marcotte column 12 line 7 through column 13, line 32.

Applicants' dependent claim 21 adds to claim 18 the limitations of "the network file server computer maintaining a generation count for each read of a file block from the file system in the disk storage in response to a read request for a file block that is not in the file system cache, and checking whether a file block having been read from the file system in the disk storage has become stale by checking whether the generation count for the file block having been

read from the file system is the same as the generation count for the last read request for the same file block." Page 15 of the Official Action again cites Marcotte column 12 line 7 through column 13, line 32. However, it is not understood how the applicants' specific claim limitations are disclosed in Marcotte column 12 line 7 through column 13, line 32.

With reference to applicants' independent claim 25, page 17 of the Official Action recognizes that Burns fails to explicitly recite the network file server responding to concurrent write requests by writing new data for specified blocks of the file to disk storage without writing the new data for the specified blocks of the file to the file system cache, and invalidating the specified blocks of the file in the file system cache. (See, e.g., applicants' FIG. 10, steps 515 and 516; applicants' spec., page 23 lines 19-23 and page 24 lines 7-15.) Page 17 of the Official Action further recognizes that Burns fails to explicitly recite the network file server responding to read requests for file blocks not found in the file system cache by reading the file blocks from the file system in disk storage and then checking whether the file blocks have become stale due to concurrent writes to the file blocks, and writing to the file system cache a file block that has not become state, and not writing to the file system cache a file block that has become stale. (See, e.g., applicants' FIG. 10, steps 92, 97, 513, 98; applicants' spec., page 23 lines 5-17; page 24 lines 16-22.) Page 18 of the Official Action cites Marcotte col. 12 line 7 through column 12, line 32 for all of these limitations not explicitly recited in Burns. However, it is not understood how all of the applicants' specific claim limitations are disclosed in Marcotte column 12 line 7 through column 13, line 32.

Marcotte column 12 line 7 through column 13, line 32, deals generally with maintaining a list of lock waiters. As shown in Marcotte FIG. 9, when a task waits for a lock, it queues its

WAIT_ELEMENT to the lock using lock or wait routine 175 and also adds it WAIT_ELEMENT to a global list or queue 230 before it waits, and removes it from the global list 230 after it waits. As shown in Marcotte FIG. 11, do_wait 240 is executed each time a thread needs to go into a wait for a lock. Step 241 executes a lock_UpdateResource procedure. Step 242 waits on ecb; and upon receiving it, step 243 executes the lock_Update Resource procedure. Marcotte says that in this way, the task waiting on a lock 100 will only actually suspend itself on the WAIT call. Adding and removing tasks (WAIT ELEMENTS) from global list 230 is done in a manner guaranteed to make the calling task wait. FIG. 12 shows that an add_to_global routine 245 (with waitp-arg) includes step 246 which determines if prevent flag is null; if so, step 248 posts an error code in the ecb field 108 of the WAIT_ELEMENT being processed; and, if not, step 247 adds the WAIT_ELEMENT (in this case, task 232) to the head of global list 230. FIG. 13 shows a remove_from_global routine 250 (with waitp=arg) including step 251 which determines if prevent flag 124 is null. If so, return code (rc) is set to zero; and if not, this WAIT_ELEMENT (say, 233) is removed from global list 230. In step 254, the return code (rc) is returned to the caller. FIG. 14 shows a return_wait routine 260 (with waitp=prc) including step 261 which determines if waitp is null. If not, the WAIT_ELEMENT pointed to by waitp is returned to free storage. Return wait 260 is the post processing routine for remove_from_global 250, and remove_from_global communicates the address of the WAIT_ELEMENT via its return code, that is input to return_wait (automatically by the resource update facility) as prc. Return_wait 260 returns the WAIT_ELEMENT to free storage. Since routine 260 is a post processing routine, the free storage return is NOT performed while holding the lock. This shows the benefits of a post processing routine, and passing the return value from a resource update routine to the post

processing routine. FIG. 15 shows quiesce_global 265 wakes up all waiters and in step 267 tells them that the program is terminating due to error by way of prevent flag 124 being set to 1 in step 266. In step 268 pointer 231 and 234 are cleared so global list 230 is empty.

Thus, it is not understood how Marcotte's general teaching of a way of maintaining a list of lock waiters discloses the applicants' specific limitations of the network file server responding to concurrent write requests by writing new data for specified blocks of the file to disk storage without writing the new data for the specified blocks of the file to the file system cache, invalidating the specified blocks of the file in the file system cache, and responding to read requests for file blocks not found in the file system cache by reading the file blocks from the file system in disk storage and then checking whether the file blocks have become stale due to concurrent writes to the file blocks, and writing to the file system cache a file block that has not become state, and not writing to the file system cache a file block that has become stale.

"[R]ejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." In re Kahn, 441 F. 3d 977, 988 (Fed. Cir. 2006). A fact finder should be aware of the distortion caused by hindsight bias and must be cautious of arguments reliant upon ex post reasoning. See KSR International Co. v. Teleflex Inc., 550 U.S. __, 82 USPQ2d 1385 (2007)), citing Graham, 383 U. S. at 36 (warning against a "temptation to read into the prior art the teachings of the invention in issue" and instructing courts to "guard against slipping into the use of hindsight.").

Applicants' dependent claim 26 adds to claim 25 the limitations of "the network file server computer checking a read-in-progress flag for a file block upon finding that the file block

is not in the file system cache, and upon finding that the read-in-progress flag indicates that a prior read of the file block is in progress from the file system in the disk storage, waiting for completion of the prior read of the file block from the file system in the disk storage, and then again checking whether the file block is in the file system cache." Page 18 of the Official Action again cites Marcotte column 12 line 7 through column 13, line 32. However, it is not understood how these specific limitations are disclosed in Marcotte column 12 line 7 through column 13, line 32.

Applicants' dependent claim 27 adds to claim 25 the limitations of "the network file server computer setting a read-in-progress flag for a file block upon finding that the file block is not in the file system cache and then beginning to read the file block from the file system in disk storage, clearing the read-in-progress flag upon writing to the file block on disk, and inspecting the read-in-progress flag to determine whether the file block has become stale due a concurrent write to the file block." Page 19 of the Official Action again cites Marcotte column 12 line 7 through column 13, line 32. However, it is not understood how these specific limitations are disclosed in Marcotte column 12 line 7 through column 13, line 32.

Applicants' dependent claim 28 adds to claim 25 the limitations of "the network file server computer maintaining a generation count for each read of a file block from the file system in the disk storage in response to a read request for a file block that is not in the file system cache, and checking whether a file block having been read from the file system in the disk storage has become stale by checking whether the generation count for the file block having been read from the file system is the same as the generation count for the last read request for the same file block." Page 19 of the Official Action again cites Marcotte column 12 line 7 through

column 13, line 32. However, it is not understood how these specific limitations are disclosed in Marcotte column 12 line 7 through column 13, line 32.

With respect to dependent claims 37-38, applicants further traverse for the reasons given above with reference to claims 5-6.

With respect to dependent claims 46-48, applicants further traverse for the reasons given above with reference to claims 12-14.

With respect to dependent claim 50, applicants further traverse for the reasons given above with reference to claim 16.

With respect to claims 51-54, applicants respectfully traverse for the reasons given above with reference to claims 25-28.

On page 20 of the Official Action, applicants' dependent claims 59 and 60 were rejected under 35 U.S.C. 103(a) as being unpatentable over Burns in view of Xu et al. U.S. Pat. 6,324,581. In reply, applicants respectfully submit that applicants' dependent claims 59 and 60 are patentable over the proposed combination of Burns and Xu due to the limitations of their independent base claim 58. Burns is distinguished from the base claim 58 as discussed above with reference to claim 1. Xu is distinguished from the base claim 58 in a similar fashion Xu fails to disclose appellants' step c) of releasing the allocation mutex of the file [prior to the step d) of issuing asynchronous write requests for writing to the file], and appellants' step f) of obtaining the allocation mutex for the file [after the step d) of issuing asynchronous write requests for writing to the file].

In view of the above, it is respectfully submitted that the application is in condition for allowance. Reconsideration and early allowance are earnestly solicited.

Respectfully submitted,

/ Richard C. Auchterlonie /

Richard C. Auchterlonie, Reg. No. 30,607
NOVAK DRUCE & QUIGG, LLP
1000 Louisiana, 53rd Floor
Houston, TX 77002
Telephone 713-571-3460
Telefax 713-456-2836
Richard.Auchterlonie@novakdruce.com